

Sesta Esercitazione

File
comandi
Unix

Esempio di file comandi

Scrivere un file comandi da invocare come segue:

./esempio D

dove D è il nome di una directory esistente.

Dopo un opportuno controllo sugli argomenti, lo script dovrà controllare ogni 5 secondi se sono stati **creati o eliminati file nella directory D**:

- In caso di cambiamento, si deve **visualizzare un messaggio su stdout** che comunichi quanti file sono presenti nella directory.

Suggerimento: uso di un file temporaneo, in cui tenere traccia del numero di file presenti al controllo precedente

Esempio: soluzione

numero di parametri, \$0 escluso

```
#!/bin/bash
if [ $# -ne 1 ] ; then echo Sintassi! ; exit; fi
if [ -d $1 ]; then echo $1 è una directory esistente
else echo $1 non è un dir!; exit; fi
echo 0 > loop.$$tmp
OK=0
while [ $OK -lt 10 ]
do
    new=`ls "$1"|wc -w`
    old=`cat loop.$$tmp`
    if [ $new -ne $old ]
    then
        echo $new > loop.$$tmp
        echo in $1 ci sono $new file
    fi
    OK=`expr $OK + 1`
    sleep 5s
done
rm loop.$$tmp
```

pid del processo in esecuzione

"" evitano problemi in caso di parametro \$1 con spazi

i nomi di file in \$1 potrebbero contenere spazi. Meglio:
new=`ls -l "\$1"|wc -l`
new=`expr \$new - 1`

Esercizio 1

Creare uno script che abbia la sintassi

./elabora S

dove S è una stringa di caratteri.

Lo script deve:

- richiedere all'utente e **leggere da standard input** il path assoluto di un file F.
- **controllare** che F sia un path assoluto e corrisponda al nome di un file esistente e leggibile.
- scrivere in un **file di output** le linee del file F che contengono almeno una occorrenza della stringa S , **ordinate in ordine lessicografico inverso**.

Il file di output sarà memorizzato nella home directory dell'utente che ha invocato lo script e dovrà avere il nome:

results_<uname>.out

dove <uname> è il nome dello USER che ha invocato lo script

Esercizio 1: suggerimenti

Lettura da standard input:

- **read var1 var2**

Le stringhe in ingresso vengono attribuite alle variabili a seconda della corrispondenza posizionale

Test di file:

- **test -f <path>** Esistenza del file. Alternativa [**-f <path>**]
- **test -d <path>** Esistenza del direttorio
- **test -r <path>** Diritto di lettura (allo stesso modo, **-w** e **-x**)

Ricerca di una stringa in un file:

- **grep** (default case sensitive) → quale opzione per comportamento case insensitive? (v. man)

Ordinamento delle linee di un file:

- **sort** → quale opzione per ordinamento inverso? (v. man)

[perchè sort e non rev?]

Redirezione I/O

- **comando > F** st. output redirezionato sul file F path(>> per append)
- **comando < F** st. input preso dal file F

L'output del grep deve essere elaborato dal sort → **piping di comandi**

Esercizio 2

Realizzare un file comandi che preveda la seguente sintassi:

cerca S M D1 D2 .. DN

dove:

- **S** è una stringa
- **M** è un intero
- **D1, D2, DN** sono nomi assoluti di directory esistenti.

Il file comandi deve:

- **controllare** il corretto passaggio degli argomenti;
- **ispezionare** il contenuto di tutte le directory date (**D1, D2, .. DN**) allo scopo di **individuare tutti file che contengono esattamente M occorrenze della stringa S**.
- Il file comandi dovrà **stampare** a video il nome assoluto di ogni file con tali caratteristiche e, al termine, stampare il numero totale dei file individuati.

Esercizio 2: suggerimenti

Ciclo su un elenco di directory con path assoluto:

```
for dir in /path/to/dir1 /path/to/dir2 /path/to/dir3
do
    # do something on $dir
done
```

L'esercizio richiede di iterare su un elenco di directory fornite dalla linea di comando: quale **variabile notevole** devo utilizzare?

Come calcolare il numero di occorrenze di una stringa in un file?
Vedere **grep -o ...** e **wc -w** (consultare il man)

Altri suggerimenti

Provare i comandi a linea di comando prima di scriverli nello script bash!

posso provare i comandi semplici:

```
studente@debian:~$ grep -c file1.txt
```

ma anche i comandi più complessi come condizioni, if e cicli:

```
studente@debian:~$ if test -f pippo ; then echo  
yes ; else echo no; fi
```

```
studente@debian:~$ for fname in *; do echo  
$fname ; done
```

**Ulteriore esercizio
per continuare a casa..**

Esercizio 3

Creare uno script che abbia la sintassi

```
./conteggio F USER filedir
```

Dove: **F** è il nome relativo di un file esistente, **USER** è uno username, **filedir** è il nome assoluto di un file leggibile esistente contenente una serie di nomi assoluti di directory esistenti.

Lo script dovrà cercare **nelle directory elencate in filedir tutti file di nome F di proprietà dell'utente U**; per ogni file che soddisfa questa condizione, lo script dovrà calcolarne la dimensione in bytes e stampare la stringa seguente:

«Il file <nome file> dell'utente U nella directory <Di> contiene <dim> caratteri.»

Esercizio 3: suggerimenti

Ciclo su un elenco di directory contenute in un file:

- ricordiamo che il comando `cat` stampa il contenuto del file dato come arg.
- ricordiamo il significato dei backquote: ``cat FILEDIR``

Come estrarre l'username del proprietario di un file?

- ricordiamo il comando `cut` applicato all'output di `ls -l`

```
-rw-r--r--  1 anna staff  2717 15 Apr 10:16 esempio5.c
```

lo username è il terzo "campo" → `cut` con opzione `-f3`
